TP1 MI11/AI39 Linux Xenomai

Semestre printemps 2025

Guillaume Sanahuja – guillaume.sanahuja@hds.utc.fr

Table des matières

Préambule

Pour ce TP, vous devez réutiliser la machine virtuelle de la séance précédente.

Un compte rendu de TP au format PDF est à rendre sur le moodle de l'UV. Il vous est demandé d'y écrire les réponses aux questions figurant dans les différents exercices. Également, vous pouvez compléter le compte rendu avec toute information que vous jugerez utile, par exemple les manipulations que vous avez réalisées pendant la séance, les résultats que vous avez observés, etc.

Travail préalable à la séance de TP

Relire les cours sur Linux embarqué et sur Linux temps réel disponibles sur le moodle MI11. Regarder la documentation de Xenomai :
https://doc.xenomai.org/v3/html/xeno3prm/index.html
Et lire notamment la documentation sur l'API alchemy, que vous utiliserez pour ce TP :
https://doc.xenomai.org/v3/html/xeno3prm/group__alchemy.html

Exercice 1 : Mise en place de Xenomai

Lors des séances précédentes, nous n'avions pas utilisé Xenomai. Il faut donc utiliser *bitbake* pour faire un noyau et une image avec le support de Xenomai. La toolchain par contre n'a pas besoin d'être refaite.

La MACHINE à utiliser dans la configuration de Yocto est « joypinote-xenomai ».

Question 1.1 : Indiquez les étapes pour générer une core-image-base avec cette nouvelle MACHINE.

Question 1.2 : Indiquez les étapes pour que la cible puisse démarrer sur les fichiers générés (bootloader, noyau, configuration du bootloader et du noyau, file system).

Vous pouvez vérifier le bon fonctionnement de Xenomai en lançant le programme *latency* sur la cible.

Exercice 2 : Tâches

Dans cet exercice, nous allons créer une application simple de type « Hello World » sous Xenomai. L'objectif principal sera de manipuler les tâches temps réel et d'analyser leur fonctionnement.

Reprenez le code du programme « Hello World » du TP sur Linux embarqué et adaptez le pour afficher le message à intervalle régulier (une fois par seconde par exemple). Cross-compilez ce code et exécutez-le sur la cible.

Question 2.1 : Ce code s'exécute-t-il de façon temps réel ? Comment le vérifier (regardez le fichier de statistiques de Xenomai)?

Créez maintenant une tâche temps réel Xenomai (avec l'API alchemy) qui se chargera d'exécuter le *printf* et le *sleep*. Afin de cross-compiler ce programme et les suivants, vous devez utiliser le *Makefile* disponible sur le moodle de l'UV. Dans ce *Makefile*, vous aurez à adapter la ligne :

EXECUTABLE := helloworld

Et changer *helloworld* par le nom de votre programme.

Question 2.2 : Donnez le code du programme. Est-il vraiment temps réel et pourquoi? Que donne le fichier de statistiques de Xenomai (ne pas interpréter ce résultat pour le moment)?

Remplacez maintenant l'appel à la fonction *sleep* par son équivalent sous Xenomai. Sous Xenomai, les temps sont exprimés en ns.

Question 2.3 :Donnez le code du programme et les statistiques de Xenomai (ne pas interpréter ce résultat pour le moment).

Remplacez maintenant l'appel à la fonction *printf* par son équivalent sous Xenomai.

Question 2.4 :Donnez le code du programme et les statistiques de Xenomai. Interprétez maintenant les résultats des différentes statistiques que vous avez relevées.

Exercice 3 : Synchronisation

Créez un programme lançant deux tâches Xenomai qui afficheront chacune une partie d'un message (chaque tâche ne doit rien faire d'autre).

Question 3.1 : Donnez le code du programme et le résultat.

Question 3.2 : Quelle est l'influence de la priorité des tâches ? Comment faire pour afficher le message dans l'ordre ou le désordre ? Justifiez.

Afin d'améliorer le comportement de votre programme, utilisez un sémaphore qui bloquera chacune des tâches dès le début. Les tâches devront être libérées après avoir été toutes lancées.

Question 3.3 : A quelle valeur faut-il initialiser le sémaphore ?

Question 3.4 : Quelle est l'influence du paramètre mode *utilisé à la création du sémaphore* sur votre programme?

Question 3.5 : Donnez le code du programme et le résultat. Expliquez le fonctionnement du programme et justifiez vos choix de priorités pour les tâches et de mode pour le sémaphore.

Nous allons maintenant faire l'affichage du message en boucle (une fois par seconde), grâce à une troisième tâche qui servira de métronome :

- modifiez les deux tâches d'affichage pour faire l'affichage en boucle
- ajoutez la troisième tâche qui synchronisera les deux autres grâce à un sémaphore et réalisera l'attente

Question 3.6 : Donnez le code du programme et le résultat. Expliquez le fonctionnement du programme et justifiez vos choix de priorités pour les tâches et de mode pour le sémaphore.

Question 3.7 : Regardez le fichier de statistiques et du scheduler de xenomai. Quelles sont les différentes informations ?

Exercice 4 : Latence

Dans cet exercice, nous allons nous intéresser à la latence de Xenomai, et la comparer avec les résultats du TP précédent.
Il faut donc réécrire un programme réalisant 10 000 fois une attente de 1ms et affichant le temps total d'attente, en utilisant Xenomai.
Xenomai ayant un système pour anticiper la latence, il faut d'abord le désactiver pour ne pas fausser les résultats que vous allez obtenir :

echo 0 > /proc/xenomai/latency

Question 4.1 : Donnez le code du programme et le résultat.

Ajoutez la mesure des latences minimum, maximum et moyenne. Veillez à utiliser les fonctions Xenomai pour la prise de temps et à utiliser le ficher entête correspondant.

Question 4.2 : Donnez le code du programme et les résultats obtenus. Que pouvez vous en conclure ?

Question 4.3 : Chargez le CPU et donnez les résultats obtenus. Que pouvez vous en conclure ?